USO DO GCC

Aluno: Paulo Filipe Moreira da Silva

Saída do arquivo 2.in antes da identificação dos problemas e aplicação das correções:

```
Inicializando matriz A[n_linhas=2][n_colunas=2].
Definido A[1][1] = 100 .
Definido A[0][1] = 200 .
Definido A[1][0] = 300 .
Definido A[0][0] = 400 .
A=(nlin=2,ncol=2,[0,1]=200,[0,0]=400,[1,1]=100,[1,0]=300)
Definido A[1][0] = 500 .
A=(nlin=2,ncol=2,[0,1]=200,[0,0]=400,[1,1]=100,[1,0]=300,[1,0]=500)
```

Saída do arquivo 2.in após da identificação dos problemas e aplicação das correções:

```
Inicializando matriz A[n_linhas=2][n_colunas=2].
Definido A[1][1] = 100 .
Definido A[0][1] = 200 .
Definido A[1][0] = 300 .
Definido A[0][0] = 400 .
A=(nlin=2,ncol=2,[0,0]=400,[0,1]=200,[1,0]=300,[1,1]=100)
Definido A[1][0] = 500 .
A=(nlin=2,ncol=2,[0,0]=400,[0,1]=200,[1,0]=500,[1,1]=100)
```

Os erros identificados estão nas linhas 7, 28 e 89:

```
7    while(atual_linha!= ap_mat -> cabeca && atual_linha->linha <=nlin){   // ERRO
8        atual_linha = atual_linha -> abaixo;
9    }
```

```
28   while(atual_coluna!= ap_mat -> cabeca && atual_coluna->coluna <=ncol){   // ERRO
29       atual_coluna = atual_coluna -> direita;
30   }
```

```
89   while(atual_linha->coluna <= col && atual_linha != cab_lin){   // ERRO
90       atual_linha = atual_linha -> direita;
91   }
```

Correções:

```
7    while(atual_linha!= ap_mat -> cabeca && atual_linha->linha < nlin){ // CORREÇÃO
8        atual_linha = atual_linha -> abaixo;
9    }
```

```
28    while(atual_coluna!= ap_mat -> cabeca && atual_coluna->coluna < ncol){  // CORREÇÃO
29        atual_coluna = atual_coluna -> direita;
30    }
```

```
89    while(atual_linha->coluna < col && atual_linha != cab_lin){  // CORREÇÃO
90        atual_linha = atual_linha ->  direita;
91    }
```

Os comandos do GDB utilizados foram “p” e “n”.

O problema era que, em todos os casos, se desejava “avançar” até chegar em uma posição específica, portanto, deve-se utilizar um código parecido com o seguinte:

```
while(percorre->valor < valor_de_parada->valor){
  percorre = percorre->proximo;}
```

Ou seja, enquanto a variável que percorrerá os elementos for menor que o ***valor_de_parada*** (seja o valor específico e/ou nó cabeça), a mesma continuará a receber o valor seguinte ao atual. Entretanto, o código implementado era parecido com o seguinte:

```
while(percorre->valor <= valor_de_parada->valor){
  percorre = percorre->proximo;}
```

A diferença é que o uso do operador “<=” implica que mesmo quando o ***valor_de_parada*** for atingido, a variável ***percorre*** receberá o próximo valor, e só então o ***while*** será encerrado.